Arbitration For Data Integrity in MC/ServiceGuard Clusters

Manufacturing Part Number : B3936-90070

July 2002

Legal Notices

Arbitration for Data Integrity in MC/ServiceGuard Clusters

Clustering is an architecture of interconnected servers that allows multiple hosts to run the same applications, permitting the individual systems to be up or down. Applications move easily between systems, accessing the same shared data from different nodes at different times. The goal is to provide high availability for the application and the data without endangering the integrity of the data. Particularly, the cluster manager must ensure that when an application is running and processing data on one node in the cluster, the same application does not inappropriately start up on another node and begin processing the same data at the same time. Cluster software must protect data integrity at all costs.

Different clustering solutions have adopted various methods for ensuring the safety of data. Some solutions rely solely on redundant membership links to ensure data integrity. Others, including the MC/ServiceGuard product family, use a process called **arbitration** to keep more than one incarnation of a cluster from running and starting up a new instance of an application. The MC/ServiceGuard team has always agreed that definitive arbitration should be used in high availability systems that run mission-critical applications.

This paper describes some basic cluster membership concepts, then discusses some different ways of determining cluster membership. Several types of arbitration are shown with example configurations that illustrate each type. Then the arbitration methods used by MC/ServiceGuard and related products are presented in some detail.

NOTE Some of the arbitration methods mentioned in this paper are available only on particular platforms or with specific types of configuration. Other methods, while possible, have not been fully tested on all types of systems. For details about support of particular kinds of arbitration, refer to your platform's *Configuration Guide*, available through your HP representative.

Cluster Membership Concepts

What is arbitration? Why is it necessary? When and how is it carried out? To answer these questions, it is necessary to explain a number of clustering concepts that are central to the processes of cluster formation and re-formation. These concepts are **membership**, **quorum**, **split-brain**, and **tie-breaking**.

Membership

A cluster is a networked collection of nodes. The key to success in controlling the location of applications in the cluster and ensuring there is no inappropriate duplication is maintaining a well-defined cluster node list. When the cluster starts up, all the nodes communicate and build this membership list, a copy of which is in the memory of every node. The list is validated continuously as the cluster runs; this is done by means of heartbeat messages that are transmitted among all the nodes. As nodes enter and leave the cluster, the list is changed in memory. Changes in membership can result from an operator's issuing a command to run or halt a node, or from system events that cause a node to halt, reboot, or crash. Some of these events are routine, and some may be unexpected. There are frequent cases in cluster operation when cluster membership is changing and when the cluster software must determine which node in the cluster should run an application.

How does the cluster software tell where an application should run? In a running cluster, when one system cannot communicate with the others for a significant amount of time, there can be several possible reasons:

1. The node has crashed.
2. The node is experiencing a kernel hang, and processing has stopped.
3. The cluster is partitioned because of a network problem. Either all the network cards connecting the node to the rest of the cluster have failed, or all the cables connecting the cards to the network have failed, or there has been a failure of the network itself.

It is often impossible for the cluster manager software to distinguish (1) from (2) and (3), and therein lies a problem, because in case (1), it is safe to restart the application on another node in the cluster, but in (2) and (3), it is not safe.

When the cluster is part of a disaster tolerant solution that has nodes located in more than one data center, loss of communication can easily happen unless redundant networking is implemented with different routing for the redundant links.

In all the above cases, the loss of heartbeat communication with other nodes in the cluster causes the re-formation protocol to be carried out. This means that nodes attempt to communicate with one another to rebuild the membership list. In case (1) above, the running nodes choose a coordinator and re-form the cluster with one less node. But in case (3), there are two sets of running nodes, and the nodes in each set attempt to communicate with the other nodes in the same set to rebuild the membership list. The result is that the two sets of nodes build different lists for membership in the new cluster. Now, if both sets of nodes were allowed to re-form the cluster, there would be two instances of the same cluster running in two locations. In this situation, the same application could start up in two different places and modify data inappropriately. This is an example of data corruption.

How does MC/ServiceGuard handle cases like the above partitioning of the cluster? The process is called **arbitration**. In the MC/ServiceGuard user's manual, the process is known as tie-breaking, because it is a means to decide on a definitive cluster membership when different competing groups of cluster nodes are independently trying to re-form a cluster.

At cluster startup time, nodes join the cluster, and a tally of the cluster membership is created and maintained in memory on all cluster nodes. Occasionally, changes in membership occur. For example, when the administrator halts a node, the node leaves the cluster, and the cluster membership data in memory is changed accordingly.

When a node crashes, the other nodes become aware of this by the fact that no cluster heartbeat is received from that node after the expected interval. Thus, the transmission and receipt of heartbeat messages is essential for keeping the membership data continuously up-to-date. Why is this membership data important? In MC/ServiceGuard, a basic package, containing an application and its data, can only be allowed to run on one node at a time. Therefore, the cluster needs to know what nodes are running in order to tell whether it is appropriate or not to start a package, and where the packages should be started. A package should not be started if it is already running; it should be started on an alternate node if the primary node is down; and so forth.

Quorum

Cluster re-formation takes place when there is some change in the cluster membership. In general, the algorithm for cluster re-formation requires the new cluster to achieve a **cluster quorum** of a strict majority (that is, more than 50%) of the nodes previously running. If both halves (exactly 50%) of a previously running cluster were allowed to re-form, there would be a **split-brain** situation in which two instances of the same cluster were running.

Split-Brain

How could a split-brain situation arise? Suppose a two-node cluster experiences the loss of all network connections between the nodes. This means that cluster heartbeat ceases. Each node will then try to re-form the cluster separately. If this were allowed to occur, it would have the potential to run the same application in two different locations and to corrupt application data. In a split-brain scenario, different incarnations of an application could end up simultaneously accessing the same disks. One incarnation might well be initiating recovery activity while the other is modifying the state of the disks. MC/ServiceGuard's quorum requirement is designed to prevent a split-brain situation.

How likely is a split-brain situation? Partly, the answer to this depends on the types of intra-node communication the cluster is using: some types are more robust than others. For example, the use of the older coaxial cable technology makes communication loss a significant problem. In that technology, the loss of termination would frequently result in the loss of an entire LAN. On the other hand, the use of redundant groups of current Ethernet hubs makes the loss of communication between nodes extremely unlikely, but it is still possible. In general, with mission-critical data, it is worth the cost to eliminate even small risks associated with split-brain scenarios.

A split-brain situation is more likely to occur in a two-node cluster than in a larger local cluster that splits into two even-sized sub-groups. Split-brain is also more likely to occur in a disaster-tolerant cluster where separate groups of nodes are located in different data centers.

Tie-Breaking

Tie-breaking (arbitration) is only required when a failure could result in two equal-sized subsets of cluster nodes each trying to re-form the cluster at the same time. These competitors are essentially tied in the contest for the cluster’s identity. The tie-breaker selects a winner, and the other nodes leave the cluster.

Tie-breaking is done using several techniques in MC/ServiceGuard clusters:

- Through a cluster lock disk, which must be accessed during the arbitration process. (This type of arbitration is available only on HP-UX systems. MC/ServiceGuard Linux clusters do not use lock disks.)
- Through arbitrator nodes, which provide tie-breaking when an entire site fails, as in a disaster scenario. Arbitrator nodes are cluster members located in a separate data center whose main function is to increase the cluster size so that an equal partition of nodes is unlikely between production data centers.
- Through a quorum server, for clusters of any size or type.

To Arbitrate or Not to Arbitrate

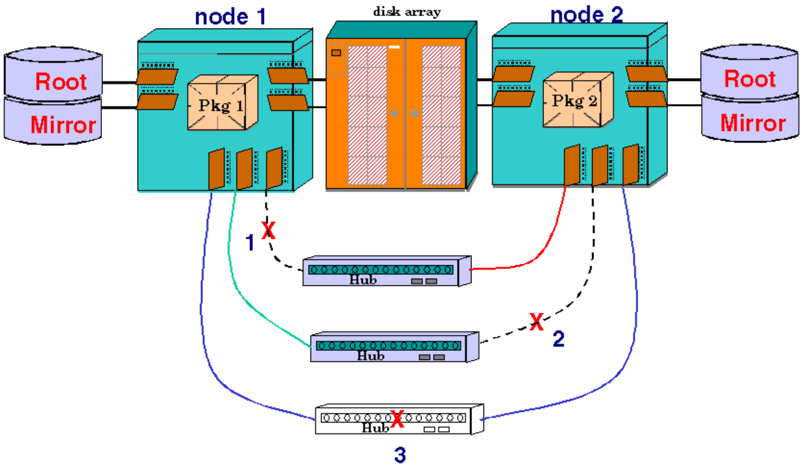
Arbitration is not always used to determine cluster membership. Some cluster software products rely exclusively on the use of multiple cluster membership communication links (heartbeats). These algorithms are described in the following sections.

No Arbitration—Multiple Paths

Some approaches do not use arbitration, but instead rely on multiple membership paths to ensure that the heartbeat or essential intra-cluster communication remains unbroken. In this approach, the event of a node failing entirely is considered more likely than the event of several LAN paths all failing at the same time. Such systems assume that a loss of communication means a node failure, and packages are allowed to fail over when a loss of heartbeat is detected.

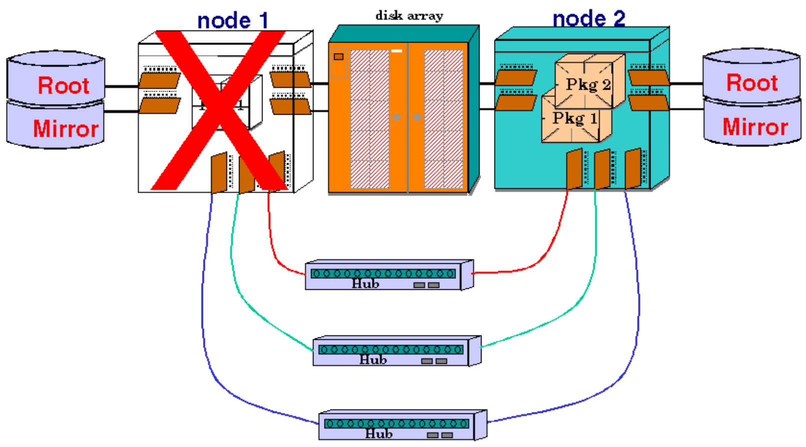
This model is illustrated in Figure 1 and Figure 2. In Figure 1, three separate LAN failures would be required to break communication between the cluster nodes. This assumes that hubs are separately powered, of course, and that other HA design criteria are met.

Figure 1 **Multiple Heartbeat Failures**



In Figure 2, on the other hand, a single node failure would result in the loss of heartbeat communication. In the no-arbitration model, the loss of heartbeat would be interpreted by the cluster manager as a failure of node 1, and therefore the cluster could re-form with packages failing over from node 1 to node 2.

Figure 2 **Single Node Failure**

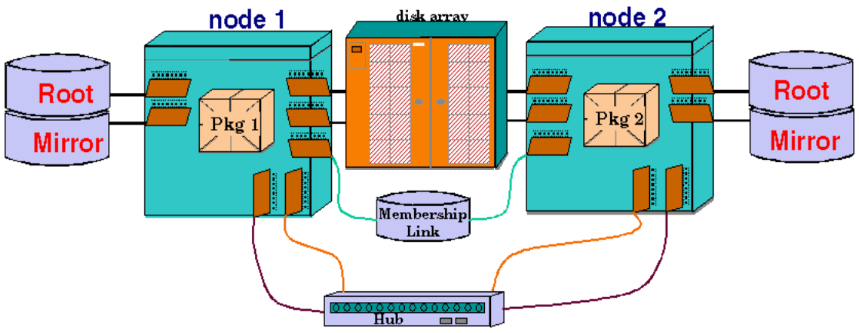


No Arbitration—Multiple Media

It is possible to define multiple membership paths for intra-cluster communication that employ different types of communication from node to node. One path could use conventional LAN links, while a second path might employ a disk.

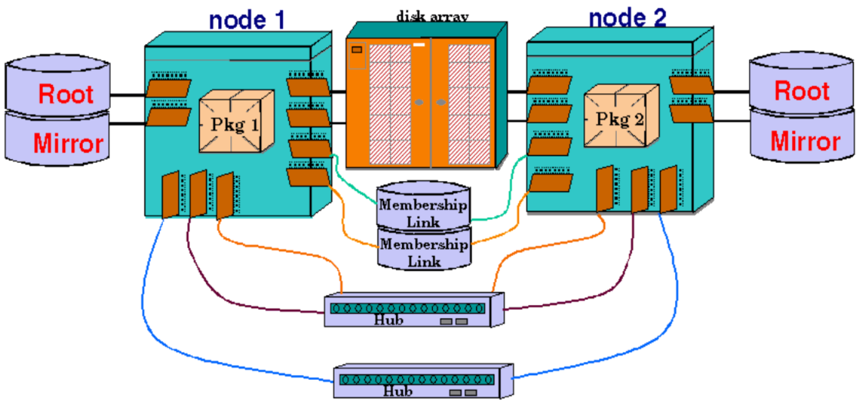
This model is illustrated in Figure 3. Both a LAN connection and a disk link provide redundant membership communication.

Figure 3 **Multiple Paths with Different Media**



Note that the configuration could be expanded to include multiple disk links plus multiple LAN links, as in Figure 4. Such a configuration would require the loss of at least 4 links for the heartbeat to be lost.

Figure 4 **Additional Multiple Paths with Different Media**



No Arbitration—Risks

When all is said and done, it may be very unlikely that intra-node communication would be lost in the above configurations, but it is still possible that heartbeat could disappear, with both nodes still running, and this scenario can cause data corruption.

The risk of split brain syndrome is considerably greater with extended distance clusters and disaster tolerant solutions in which nodes are located in different data centers at some distance from each other. For these types of solution, some form of arbitration is essential.

The unlikely but *possible* scenario of split brain can be definitively avoided with an arbitration device. In other words, the risk of data corruption can be eliminated. The HP MC/ServiceGuard family of clustering software includes this level of protection.

How MC/ServiceGuard Uses Arbitration

MC/ServiceGuard employs a lock disk, a quorum server, or arbitrator nodes to provide definitive arbitration to prevent split-brain conditions. This section describes how the software handles cluster formation and re-formation and supplies arbitration when necessary.

Cluster Startup

The **cluster manager** is used to initialize a cluster, to monitor the health of the cluster, to recognize node failure if it should occur, and to regulate the re-formation of the cluster when a node joins or leaves the cluster. The cluster manager operates as a daemon process that runs on each node. During cluster startup and re-formation activities, one node is selected to act as the **cluster coordinator**. Although all nodes perform some cluster management functions, the cluster coordinator is the central point for inter-node communication.

Startup and Re-Formation

The cluster can start when the `cmruncl` command is issued from the command line. All nodes in the cluster must be present for cluster startup to complete. If all nodes are not present, then the cluster must be started by issuing commands that specify only a specific group of nodes. This is to ensure that we do not create a split-brain situation.

Cluster re-formation occurs any time a node joins or leaves a running cluster. This can follow the reboot of an individual node, or it may be when all nodes in a cluster have failed, as when there has been an extended power failure and all SPUs went down.

Automatic cluster startup will take place if the flag `AUTOSTART_CMCLD` is set to 1 in the `/etc/rc.config.d/cmcluster` file. When any node reboots with this parameter set to 1, it will rejoin an existing cluster, or if none exists it will attempt to form a new cluster. As with the cmruncl command, automatic initial startup requires all nodes in the cluster to be present. If all nodes are not present, the cluster must be started with commands.

Dynamic Cluster Re-Formation

A dynamic re-formation is a temporary change in cluster membership that takes place as nodes join or leave a running cluster. Re-formation differs from reconfiguration, which is a permanent modification of the configuration files. Re-formation of the cluster occurs under the following conditions (not a complete list):

- An SPU or network failure was detected on an active node.
- An inactive node wants to join the cluster. The cluster manager daemon has been started on that node.
- The system administrator halted a node.
- A node halts because of a package failure.
- A node halts because of a service failure.
- Heavy network traffic prohibited the heartbeat signal from being received by the cluster.
- The heartbeat network failed, and another network is not configured to carry heartbeat.

Typically, re-formation results in a cluster with a different composition. The new cluster may contain fewer or more nodes than in the previous incarnation of the cluster.

Cluster Quorum and Cluster Locking

Recall that the algorithm for cluster re-formation requires a **cluster quorum** of a strict majority (that is, more than 50%) of the nodes previously running. If both halves (exactly 50%) of a previously running cluster were allowed to re-form, there would be a **split-brain** situation in which two instances of the same cluster were running.

Cluster Lock

Although a cluster quorum of more than 50% is generally required, MC/ServiceGuard allows exactly 50% of the previously running nodes to re-form as a new cluster *provided that the other 50% of the previously running nodes do not also re-form.* This is guaranteed by the use of an arbiter or tie-breaker to choose between the two equal-sized node groups, allowing one group to form the cluster and forcing the other group to shut down. This type of arbitration is known as a **cluster lock**.

The cluster lock is used as a tie-breaker only for situations in which a running cluster fails and, as MC/ServiceGuard attempts to form a new cluster, the cluster is split into two sub-clusters of equal size. Each sub-cluster will attempt to acquire the cluster lock. The sub-cluster which gets the cluster lock will form the new cluster, preventing the possibility of two sub-clusters running at the same time. If the two sub-clusters are of unequal size, the sub-cluster with greater than 50% of the nodes will form the new cluster, and the cluster lock is not used.

If you have a two-node cluster, you are required to configure the cluster lock. If communications are lost between these two nodes, the node that obtains the cluster lock will take over the cluster and the other node will undergo a forced halt. Without a cluster lock, a failure of either node in the cluster will result in a forced immediate system halt of the other node, and therefore the cluster will halt.

If the cluster lock fails or is unavailable during an attempt to acquire it, the cluster will halt. You can avoid this problem by configuring the cluster's hardware so that the cluster lock is not lost due to an event that causes a failure in another cluster component.

No Cluster Lock

Normally, you should not configure a cluster of three or fewer nodes without a cluster lock. In two-node clusters, a cluster lock is required. You may consider using no cluster lock with configurations of three or more nodes, although the decision should be affected by the fact that any cluster may require tie-breaking. For example, if one node in a three-node cluster is removed for maintenance, the cluster reforms as a two-node cluster. If a tie-breaking scenario later occurs due to a node or communication failure, the entire cluster will become unavailable.

In a cluster with four or more nodes, you may not need a cluster lock since the chance of the cluster being split into two halves of equal size is very small. However, be sure to configure your cluster to prevent the failure of exactly half the nodes at one time. For example, make sure there is no potential single point of failure such as a single LAN between equal numbers of nodes, and that you use multiple power circuits with less than half of the nodes on any single power circuit.

Cluster lock disks are *not* allowed in clusters of more than four nodes. A quorum server or arbitrator nodes may be employed with larger clusters, and this kind of arbitration is necessary for extended distance clusters and with MetroCluster configurations to provide disaster tolerance.

Lock Requirements

The cluster lock can be implemented either by means of a **lock disk** (HP-UX clusters only) or by means of a **quorum server** (both HP-UX and Linux clusters). A one-node cluster does not require a cluster lock. A two-node cluster *requires* a cluster lock. In larger clusters, the cluster lock is strongly recommended.If you have a cluster with more than four nodes, a cluster lock *disk* is not allowed, but you can use a quorum server. Therefore, if the cluster is expected to grow to more than 4 nodes, you should use a quorum server. In clusters that span several data centers, a more practical alternative may be the use of arbitrator nodes. Arbitrator nodes are not a form of cluster lock, but rather they are components that prevent the cluster from ever being partitioned into two equal-sized groups of nodes.

Use of a Lock Disk as the Cluster Lock

The cluster lock disk is a disk that can be written to by all members of the cluster. When a node obtains the cluster lock, this disk is marked so that other nodes will recognize the lock as “taken.” A lock disk may be used for clusters of up to four nodes.

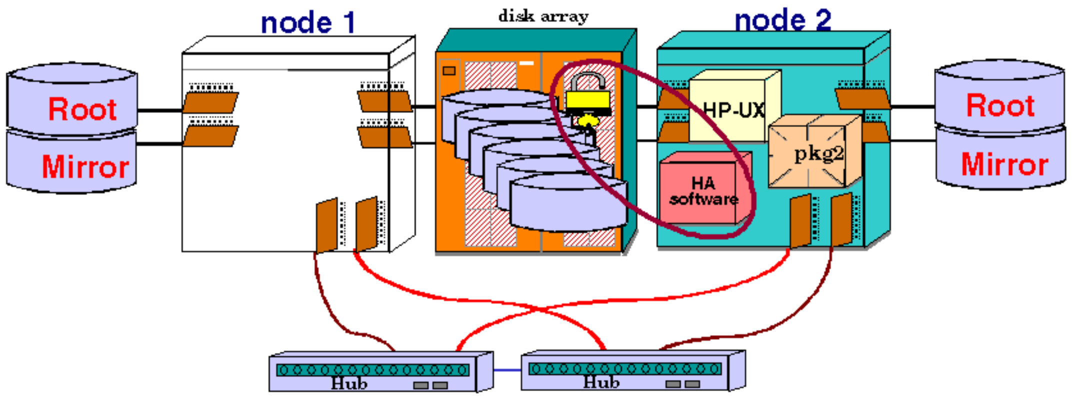
The lock is created in a special area on a particular LVM physical volume. The cluster lock volume group and physical volume names are identified in the cluster configuration file.

The lock disk is not dedicated for use as the cluster lock; thus, it can be employed as part of a normal volume group with user data on it. The usable space on the disk is not impacted; the lock disk takes no space away from the disk’s volume group. Further, the activation of the volume group on one node does not affect the ability of another node to acquire the cluster lock.

The lock area on the disk is not mirrored, even though the physical volume may be a part of a volume group that contains mirrored logical volumes.

The operation of the lock disk is shown in Figure 5. The node that acquires the lock (in this case node 2) continues running in the cluster. The other node halts.

Figure 5 **Lock Disk Operation**



ServiceGuard periodically checks the health of the lock disk and writes messages to the syslog file when a lock disk fails the health check. This file should be monitored for early detection of lock disk problems.

You can choose between two lock disk options—a single or dual lock disk—based on the kind of high availability configuration you are building. *A single lock disk is recommended where possible*. With both single and dual locks, however, it is important that the cluster lock be available even if the power circuit to one node fails; thus, the choice of a lock configuration depends partly on the number of power circuits available. Regardless of your choice, all nodes in the cluster must have access to the cluster lock to maintain high availability.

Single Cluster Lock

It is recommended that you use a single lock disk. A single lock disk should be configured on a power circuit separate from that of any node in the cluster. For example, it is highly recommended to use three power circuits for a two-node cluster, with a single, separately powered disk for the cluster lock. For two-node clusters, this single lock disk may not share a power circuit with either node, and it must be an external disk. For three or four node clusters, the disk should not share a power circuit with 50% or more of the nodes.

Dual Cluster Lock

In an extended distance cluster, where the cluster contains nodes running in two separate data centers, a single lock disk would be a single point of failure should the data center it resides in suffer a catastrophic failure. *In this case only*, a dual cluster lock, with two separately powered disks, should be used to eliminate the lock disk as a single point of failure. The use of the dual cluster lock is further shown in “Use of Dual Lock Disks in Extended Distance Clusters” on page 26.

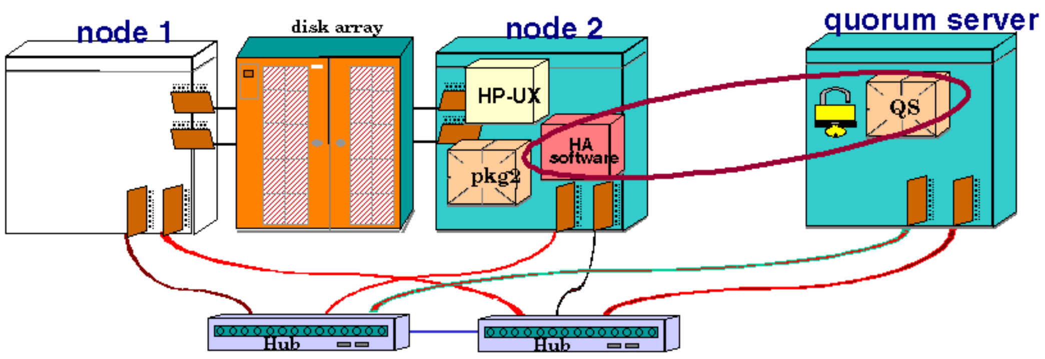
Use of a Quorum Server as the Cluster Lock

A quorum server can be used in clusters of any size. The quorum server is an alternate form of cluster lock that uses a server program running on a separate system for tie-breaking rather than a lock disk. Should two equal sized groups of nodes (exactly 50% of the cluster in each group) become separated from each other, the quorum server allows one group to achieve quorum and form the cluster, while the other group is denied quorum and cannot start a cluster.

The quorum server process runs on a machine *outside of the cluster for which it is providing quorum services*. The quorum server listens to connection requests from the MC/ServiceGuard nodes on a known port. The server maintains a special area in memory for each cluster, and when a node obtains the cluster lock, this area is marked so that other nodes will recognize the lock as “taken.”

The operation of the quorum server is shown in Figure 6. When there is a loss of communication between node 1 and node 2, the quorum server chooses one node (in this example, node 2) to continue running in the cluster. The other node halts.

Figure 6 **Quorum Server Operation**



The quorum server runs on a Linux or HP-UX system. In future releases, it is expected you will be able to configure QS as an MC/ServiceGuard package in another cluster, which must be a different cluster than the one for which it is providing quorum services. A quorum server, whether or not it is running as a package, can provide quorum services for multiple clusters.

NOTE Consult your *Configuration Guide* for up-to-the-minute information on types of supported quorum server configuration, as well as on support of quorum server as a package.

Setting up the Quorum Server

If you are using a quorum server (QS), the QS software, which has to be running during cluster configuration, must be installed on a system other than the nodes on which your cluster will be running. This can be either a single HP-UX or Linux system.

NOTE In Linux, the node on which the quorum server is running must be in the same subnet as the clusters for which it is providing services. This restriction does not apply to HP-UX quorum servers. On HP-UX, the use of Auto Port Aggregation in `LAN_MONITOR` mode is recommended for the quorum server system. In Linux, channel bonding can be employed.

Ensure that the connection with each node is independent of the heartbeat connection, so that both are not likely to fail at the same time.

Installing the Quorum Server

If you are using HP-UX, run the `swinstall` command to install product number B8467BA on the system or systems where it will be running. If you are using Linux, use the rpm command to install product T1229BA. Follow the instructions in the release notes for either product. You do not need to install the product on nodes that are simply using quorum services.

Running the Quorum Server

The quorum server must be running during the following cluster operations:

- when the `cmquerycl` command is issued.

- when the `cmapplyconf` command is issued.
- when there is a cluster re-formation.

Specifying a Quorum Server

If you will be using a quorum server, be sure to specify the `-q qshost` option with the `cmquerycl` command. Example:

```
# cmquerycl -v -n lp1 -n lp2 -q lp-qs  -C clus-lp.config
```

Cluster Configuration Template File QS Section

Edit the cluster configuration ASCII file, and include values for `QS_HOST`, `QS_POLLING_INTERVAL`, and (optionally) `QS_TIMEOUT_EXTENSION`. The following is a portion of an ASCII configuration file generated with the `cmquerycl` command using the `-q` option:

```
# **********************************************************************
# ********* HIGH AVAILABILITY CLUSTER CONFIGURATION FILE ***************
# ***** For complete details about cluster parameters and how to    ****
# ***** set them, consult the cmquerycl(1m) manpage or your manual. ****
# **********************************************************************

# Enter a name for this cluster.  This name will be used to identify the
# cluster when viewing or manipulating it.

CLUSTER_NAME            clus-lp

# Quorum Server Parameters. Use the QS_HOST, QS_POLLING_INTERVAL,
# and QS_TIMEOUT_EXTENSION parameters to define a quorum server.
# The QS_HOST is the host name or IP address of the system
# that is running the quorum server process.  The
# QS_POLLING_INTERVAL (microseconds) is the interval at which
# The optional QS_TIMEOUT_EXTENSION (microseconds) is used to increase
# the time interval after which the quorum server is marked DOWN.
#
# The default quorum server timeout is calculated from the
# ServiceGuard cluster parameters, including NODE_TIMEOUT and
# HEARTBEAT_INTERVAL. If you are experiencing quorum server
# timeouts, you can adjust these parameters, or you can include
# the QS_TIMEOUT_EXTENSION parameter.
#
# For example, to configure a quorum server running on node
# "qshost" with 120 seconds for the QS_POLLING_INTERVAL and to
# add 2 seconds to the system assigned value for the quorum server
# timeout, enter:
#
# QS_HOST qshost
# QS_POLLING_INTERVAL 120000000
# QS_TIMEOUT_EXTENSION 2000000

QS_HOST         lp-qs
QS_POLLING_INTERVAL 120000000
QS_TIMEOUT_EXTENSION 2000000
```

Quorum Server Status and State

The *status* of the quorum server can be one of the following:

- `Up`. The quorum server is active.
- `Down`. The quorum server is not active.

The *state* of the quorum server can be one of the following:

- `Running`. Quorum services are active and available.
- `Unsupported Version`. There is a version mismatch between MC/ServiceGuard and Quorum Server.
- `Access Denied`. The MC/ServiceGuard node is not authorized to access the quorum server system.
- `Unknown`. MC/ServiceGuard cannot determine whether the quorum server is up or down. This may happen when the quorum server cannot be reached from the current node.

Viewing Quorum Server Status and State

If the cluster is using a quorum server for tie-breaking services, you can use the `cmviewcl` command to display the server name, state and status following the entry for each node, as in the following excerpt from the output of `cmviewcl -v`:

```
CLUSTER      STATUS
clus-lp      up

  NODE         STATUS       STATE
  lp1          up           running

Quorum Server Status:
NAME                    STATUS       STATE
lp-qs                   up           running

...

  NODE         STATUS       STATE
  lp2          up           running

Quorum Server Status:
NAME                    STATUS       STATE
lp-qs                   up           running
```

Viewing Quorum Server System Data

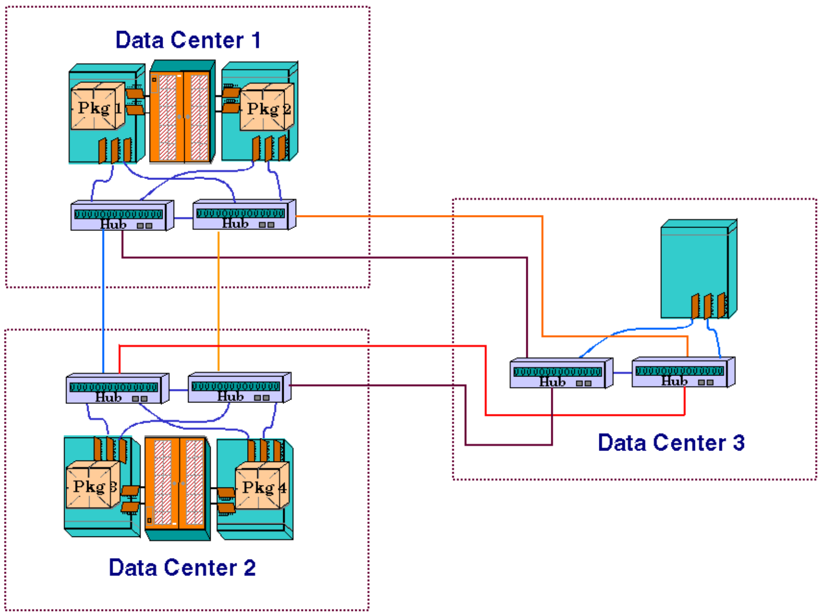
You can also log on to the quorum server system itself and use the ps command to confirm that the qs process is running. In addition, you can view data about connections from cluster nodes in the quorum server log file.

Use of Arbitrator Nodes

One way to ensure that split-brain situations do not arise is to devise an architecture that makes an even partition of the cluster impossible or at least extremely unlikely. A single failure in a four-node cluster could result in two equal-sized partitions, but a single failure in a five-node cluster could not. The fifth node in the cluster, then, performs the job of arbitration by virtue of the fact that it makes the number of nodes in the cluster odd. This type of node is sometimes called an **arbitrator node**, although it may be indistinguishable from other cluster nodes and is not configured in any special way in the cluster configuration file. This kind of arbitration is especially useful when nodes in the cluster are separated by significant distances, as in extended distance clusters or metropolitan clusters. Arbitrator nodes may be configured to run non-clustered applications, or they can be set up purely as arbitrators, with no other applications running other than MC/ServiceGuard.

The use of an arbitrator node is shown in Figure 7. The single system in Data Center 3 is an arbitrator node.

Figure 7 **Use of Arbitrator Node**



Arbitration in Disaster-Tolerant Clusters

Disaster-tolerant clusters are those which are intended to survive the loss of a data center that contains multiple resources. Examples are an extended distance cluster where nodes may be separated into different data centers on one site; metropolitan clusters, where the nodes are separated into equal-sized groups located a significant distance apart; and continental clusters, in which the geographically separate data centers provide a home for entirely separate clusters, complete with storage devices.

Extended Distance Clusters

In extended distance (campus) clusters, the nodes are divided into two separate data centers in different buildings, which can be as far as 100 km apart. A dual cluster lock disk may be used for arbitration, with the two lock disks located in the two different data centers. This affords protection against the loss of one of the data centers. If a quorum server is used, it must be in a different location from either of the two data centers, thus providing additional protection against data center loss. Similarly, if arbitrator nodes are used, they must be in a different location from the two data centers.

An extended distance cluster is no different from a standard MC/ServiceGuard cluster except that components are subdivided by data center. This means that groups of nodes are located in different buildings, and storage units with mirrored data are placed in separate facilities as well.

Metropolitan Clusters

Arbitration in a MetroCluster configuration has traditionally followed a different model than that of the single arbitrator device (originally this was a lock disk). Because a MetroCluster configuration contains two distinct data centers at some distance from one another, the main protection has been against a partition that separates the two data centers into equal-sized groups of nodes. A lock disk is not possible in this type of cluster, since metropolitan clusters use a specialized data replication method rather than LVM mirroring. Since LVM is not used with shared data, there is no one disk that is actually connected to both data centers that could act as a lock disk. Arbitration in this case can be obtained by using arbitrator nodes or a quorum server.

Arbitrator Nodes

For example, a metropolitan cluster with three nodes in Data Center A and three nodes in Data Center B could be partitioned such that two equal-sized groups remain up and running, trying to re-form. To address this problem, the supported configurations included one or two arbitrator nodes located in a third data center. These nodes are configured into the cluster for the purpose of providing a majority of nodes when combined with one half the nodes in an equal partition. In other words, if the metropolitan cluster should lose one data center, the surviving data center would still remain connected to the arbitrator nodes, so the surviving group would be larger than 50% of the previously running nodes in the cluster. It could therefore obtain the quorum and re-form the cluster.

Note that in a metropolitan cluster, it is the simple existence of the node(s) in the third data center that provides arbitration combined with the requirement that the configuration have an equal number of nodes in Data Center A and Data Center B. The arbitrator nodes located in Data Center C may do useful work, but they are not attached to the storage devices used by the main nodes in the cluster. They are fully configured as cluster nodes, but their main job is to provide arbitration.

Quorum Server

With the advent of the quorum server, another MetroCluster configuration is now possible. A quorum server process, located in a third data center, can be used for arbitration. The third data center is needed, as it was in the case of arbitrator nodes, to provide the appropriate degree of disaster tolerance. That is, the QS could arbitrate cluster re-formation if either of the other two entire sites should be destroyed.

One advantage of the quorum server is that additional cluster nodes do not have to be configured for arbitration.

Continental Clusters

There are no special arbitration requirements or configurations for the separate clusters within a continental cluster. Each cluster must provide its own arbitration separately, according to the applicable rules for a standard MC/ServiceGuard cluster. In other words, the continental cluster can employ any supported method of arbitration for its component clusters.

ContinentalClusters provides semi-automatic failover via commands which must be issued by a human operator. Between the member clusters of a continental cluster, the arbitrator is the system administrator on the recovery site, who must verify that it is appropriate to issue the `cmrecovercl` command.

Use of Dual Lock Disks in Extended Distance Clusters

Lock disks are not supported in metropolitan clusters, but they can be used in an extended distance cluster, which employs mirrored LVM over a FibreChannel disk link.

For an extended distance cluster, there should be one lock disk in each of the data centers; all nodes have access to both lock disks via the disk link. In the event of a failure of one of the data centers, the nodes in the remaining data center will be able to acquire their local lock disk, allowing them to successfully reform a new cluster. In a dual lock disk configuration, there is an extremely slight chance of split-brain in dual lock disk configuration.

The use of dual locks is as follows in an extended distance cluster:

1. After the loss of heartbeat between the data centers, each sub-cluster attempts to access the first lock disk, then the second lock disk.
2. If a node in one data center is successful at obtaining the first lock disk and the disk link between the two data centers is still viable, then nodes in the second data center will be refused the lock. The first data center will then be able to obtain the second disk and re-form the cluster.

   Note that if the first lock disk is located in the first data center when the heartbeat is lost, the first data center will normally obtain the lock first because it is closest to the disk. Thus in this scenario, the first data center will re-form the cluster.
3. If a node in one data center is successful at obtaining the first lock disk but the disk link is not viable because the other data center is down, then the first data center will not be able to obtain the second

lock disk, but *because the lock was not refused*, it will still be allowed to re-form the cluster. This is the expected behavior when there is a disaster.

4. If there is a loss of both heartbeat and disk link, there is a danger of split brain because each sub-cluster, attempting to acquire both lock disks, is able to obtain the lock in its own data center, and is *not refused* the other lock. It is important to minimize or eliminate this slight danger by ensuring that data and heartbeat links are separately routed between data centers.

NOTE

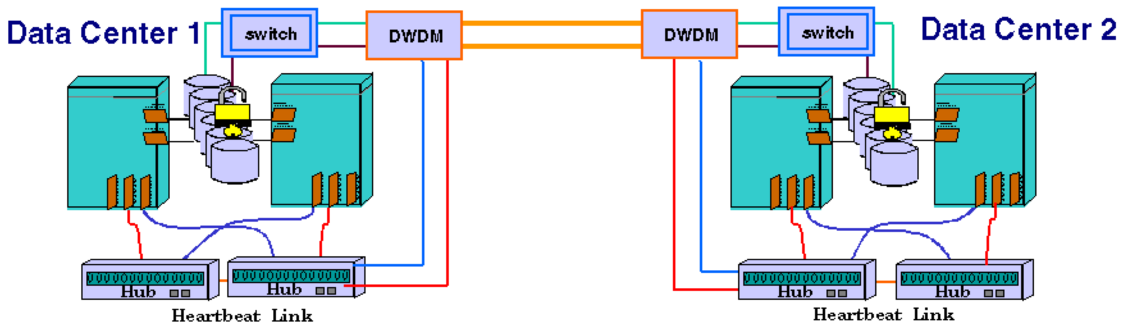
A dual lock disk configuration does not provide a redundant cluster lock. In fact, the dual lock is a *compound* lock, and both disks have to participate in the protocol of lock acquisition by the two equal-sized sets of nodes. Even when mirrored LVM is used via MirrorDisk/UX, *the lock disk area is not mirrored*.

At cluster formation time, a set of nodes must gain access to one disk, and must *either gain access to the other disk or not be denied access to it*. ("Not being denied" occurs when a disk is not accessible to a set of nodes.) The group of nodes that gains access to at least one disk and is not denied access by any disk is allowed to form the new cluster.

If one of the dual lock disks fails, MC/ServiceGuard will detect this when it carries out periodic checking, and it will write a message to the syslog file. After the loss of one of the lock disks, if the failure of a cluster node results in the need for arbitration, the cluster will go down.

A dual lock disk configuration is shown in Figure 8.

Figure 8 Dual Lock Disk Configuration



Summary

This paper has described a number of approaches to arbitration to provide safety for data in high availability clusters. There are advantages and disadvantages to each of the major approaches using a lock disk, a quorum server, or arbitrator nodes.

Table 1 Comparison of Different Arbitration Methods

Arbitration Mode	Advantages	Disadvantages
Lock Disk	• Does not require a separate system. • Disk can be used for other purposes as well. • Legacy method that has been used for many years.	• Requires separate disk hardware that must be cabled to all nodes. • Can only be used with clusters up to 4 nodes. • Not available on Linux. • Acquiring the lock follows an arbitration protocol, and this adds to failover time. • Use in extended distance clusters requires dual lock disks, with a slight risk of split brain.
Quorum Server	• Can be used with clusters of any size. • Can serve up to 50 clusters/100 nodes. • Is free on the Distributed Components CD (shipped along with the SG product CD). • Provides faster failover time than cluster lock disk.	• Requires a separate system not part of the cluster. • Only one IP address is used for quorum server. • Does not permit multiple physical network pathways unless APA (auto port aggregation) or channel bonding is used. LAN failure will remove the QS from communication with the cluster, though the cluster remains running.

Table 1 Comparison of Different Arbitration Methods (Continued)

Arbitration Mode	Advantages	Disadvantages
Arbitrator Nodes	• Offers protection for the loss of an entire data center in a metropolitan or extended distance cluster. • Allows arbitration to be done over longer distances than disk locking. • Arbitrator nodes can be used for other non-cluster applications.	• Additional MC/ServiceGuard licenses required for arbitrator nodes. • Additional hardware must be provided, often in a data center that is on a separate site. • For those with multiple clusters, arbitrator nodes can be very costly.